Lab Exercise 7– Creating Multiple IAM Users in Terraform

Objective:

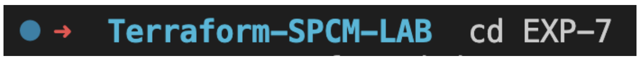
Learn how to use Terraform to create multiple IAM users with unique settings.

Prerequisites:

• Terraform installed on your machine.
• AWS CLI configured with the necessary credentials.

Steps:

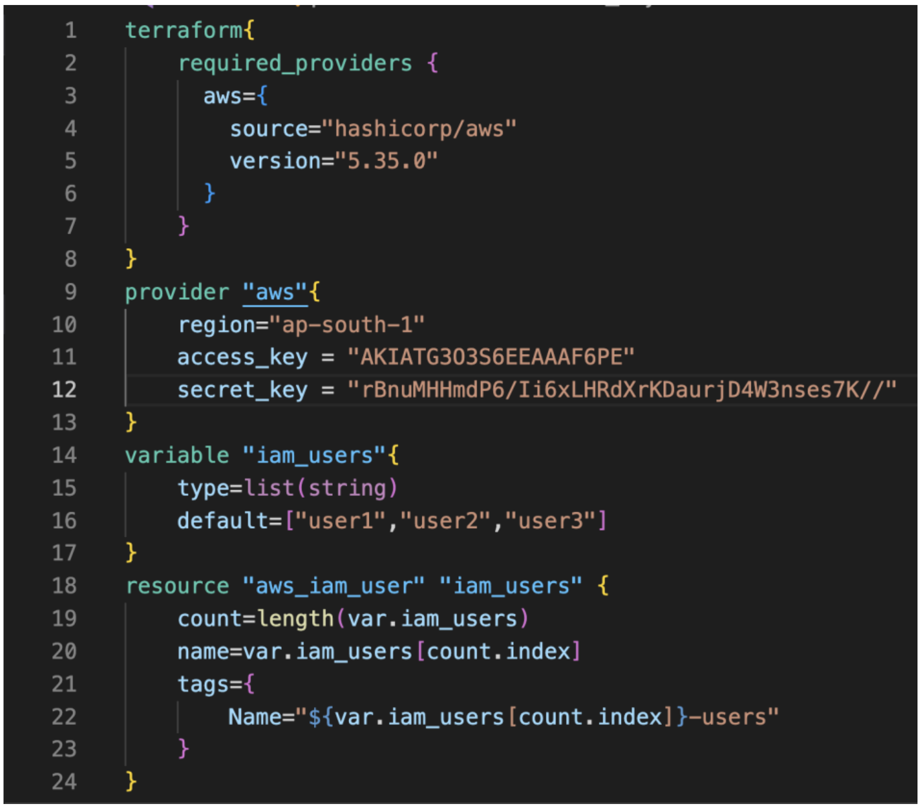
1. **Create a Terraform Directory:**



```
● ➜  Terraform-SPCM-LAB  cd EXP-7
```

• Create Terraform Configuration Files:
• Create a file named main.tf:

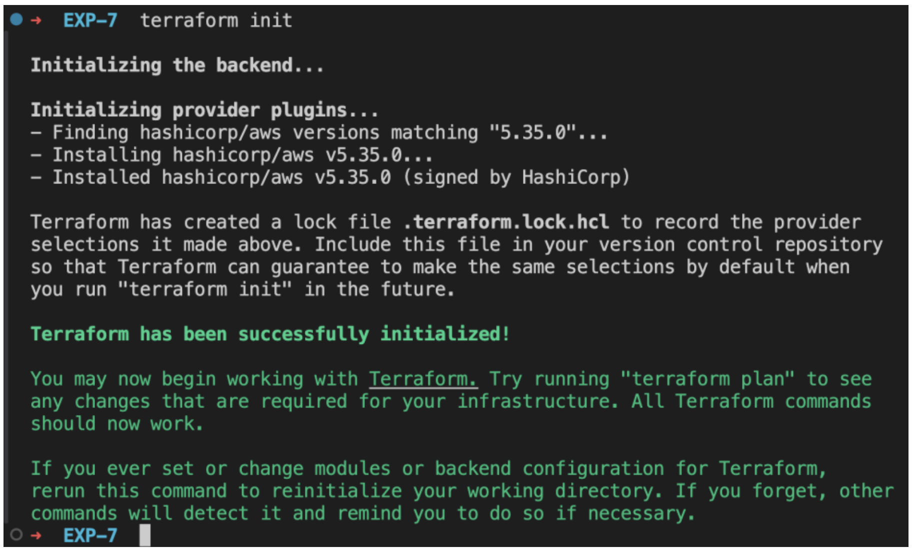
# main.tf



```
terraform{
    required_providers {
      aws={
        source="hashicorp/aws"
        version="5.35.0"
      }
    }
}
provider "aws"{
    region="ap-south-1"
    access_key = "AKIATG3O3S6EEAAAF6PE"
    secret_key = "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"
}
variable "iam_users"{
    type=list(string)
    default=["user1","user2","user3"]
}
resource "aws_iam_user" "iam_users" {
    count=length(var.iam_users)
    name=var.iam_users[count.index]
    tags={
        Name="${var.iam_users[count.index]}-users"
    }
}
```

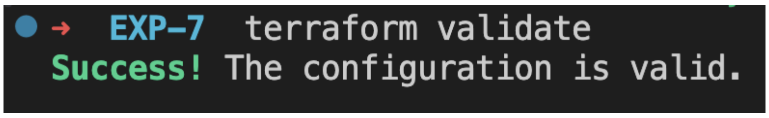
In this configuration, we define a list variable iam_users containing the names of the IAM users we want to create. The aws_iam_user resource is then used in a loop to create users based on the values in the list.

2. **Initialize and Apply:**
   Run the following Terraform commands to initialize and apply the configuration:



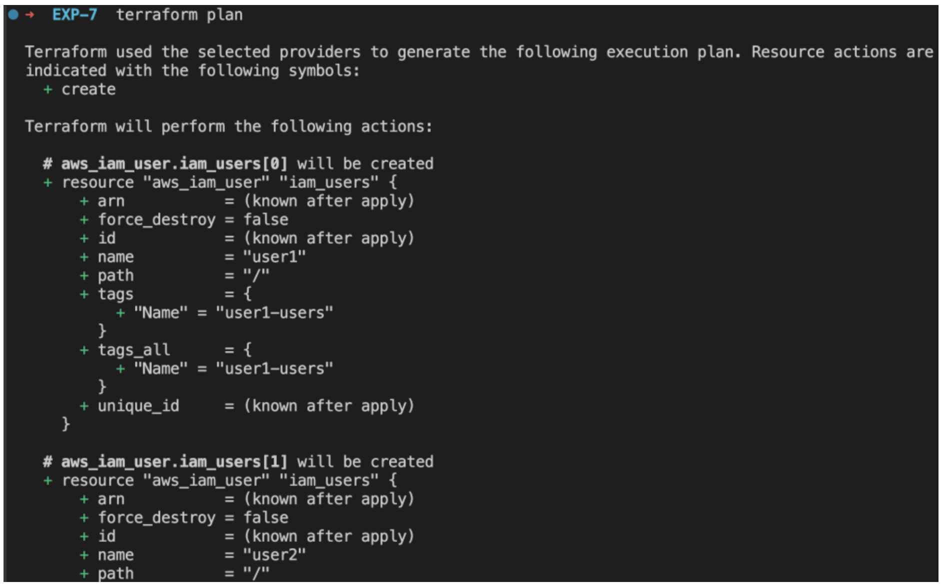
```
● → EXP-7  terraform init

  Initializing the backend...

  Initializing provider plugins...
  - Finding hashicorp/aws versions matching "5.35.0"...
  - Installing hashicorp/aws v5.35.0...
  - Installed hashicorp/aws v5.35.0 (signed by HashiCorp)

  Terraform has created a lock file .terraform.lock.hcl to record the provider
  selections it made above. Include this file in your version control repository
  so that Terraform can guarantee to make the same selections by default when
  you run "terraform init" in the future.

  Terraform has been successfully initialized!

  You may now begin working with Terraform. Try running "terraform plan" to see
  any changes that are required for your infrastructure. All Terraform commands
  should now work.

  If you ever set or change modules or backend configuration for Terraform,
  rerun this command to reinitialize your working directory. If you forget, other
  commands will detect it and remind you to do so if necessary.
○ → EXP-7
```

3. **Use terraform validate to check any error in HCL script.**



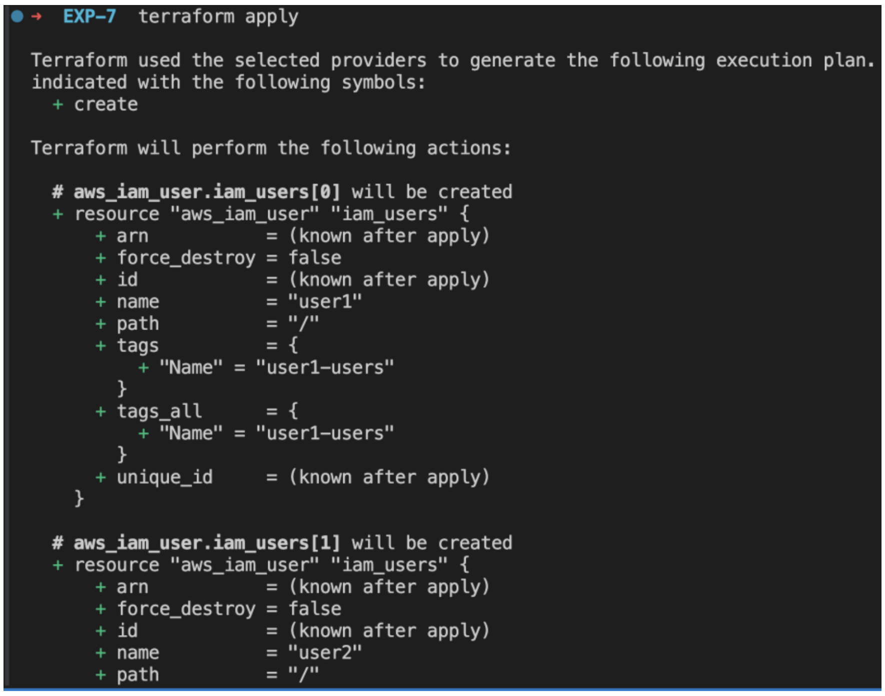
```
● → EXP-7  terraform validate
  Success! The configuration is valid.
```

4. **Use terraform plan to review the provided resources.**

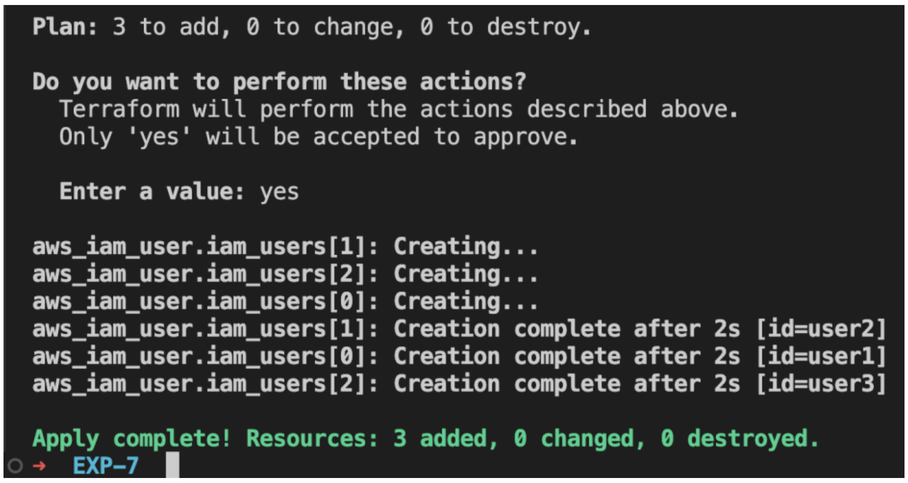


```
● → EXP-7  terraform plan

  Terraform used the selected providers to generate the following execution plan. Resource actions are
  indicated with the following symbols:
    + create

  Terraform will perform the following actions:

    # aws_iam_user.iam_users[0] will be created
    + resource "aws_iam_user" "iam_users" {
        + arn           = (known after apply)
        + force_destroy = false
        + id            = (known after apply)
        + name          = "user1"
        + path          = "/"
        + tags          = {
            + "Name" = "user1-users"
          }
        + tags_all      = {
            + "Name" = "user1-users"
          }
        + unique_id     = (known after apply)
      }

    # aws_iam_user.iam_users[1] will be created
    + resource "aws_iam_user" "iam_users" {
        + arn           = (known after apply)
        + force_destroy = false
        + id            = (known after apply)
        + name          = "user2"
        + path          = "/"
```

5. Use terraform apply to apply all the changes.

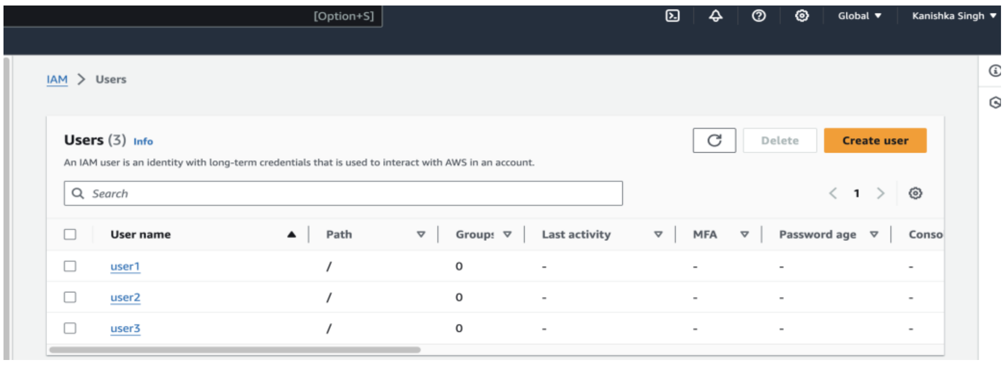


```
● ➜  EXP-7  terraform apply

  Terraform used the selected providers to generate the following execution plan.
  indicated with the following symbols:
    + create

  Terraform will perform the following actions:

    # aws_iam_user.iam_users[0] will be created
    + resource "aws_iam_user" "iam_users" {
        + arn           = (known after apply)
        + force_destroy = false
        + id            = (known after apply)
        + name          = "user1"
        + path          = "/"
        + tags          = {
            + "Name" = "user1-users"
          }
        + tags_all      = {
            + "Name" = "user1-users"
          }
        + unique_id     = (known after apply)
      }

    # aws_iam_user.iam_users[1] will be created
    + resource "aws_iam_user" "iam_users" {
        + arn           = (known after apply)
        + force_destroy = false
        + id            = (known after apply)
        + name          = "user2"
        + path          = "/"
```



```
Plan: 3 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_iam_user.iam_users[1]: Creating...
aws_iam_user.iam_users[2]: Creating...
aws_iam_user.iam_users[0]: Creating...
aws_iam_user.iam_users[1]: Creation complete after 2s [id=user2]
aws_iam_user.iam_users[0]: Creation complete after 2s [id=user1]
aws_iam_user.iam_users[2]: Creation complete after 2s [id=user3]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
○ ➜  EXP-7
```

6. Verify using AWS console.

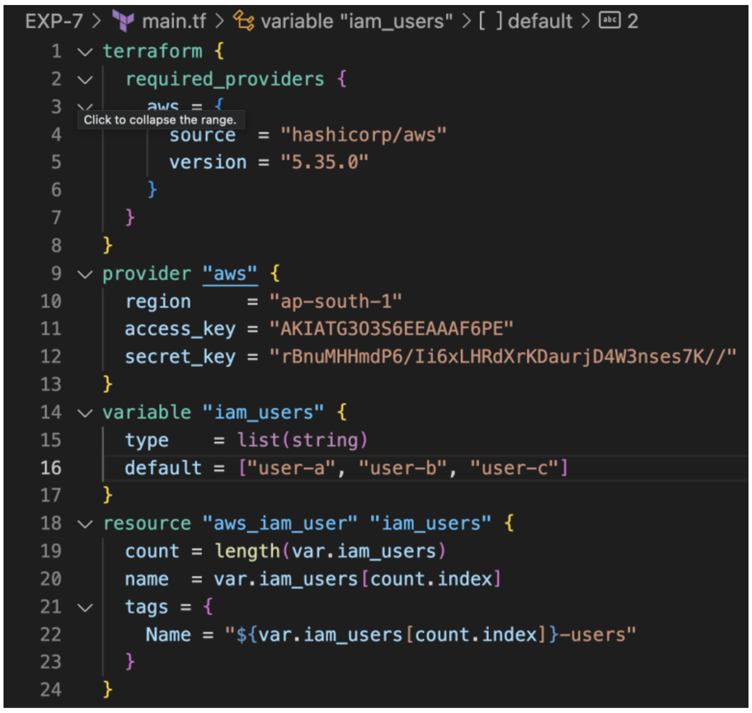


Users (3) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

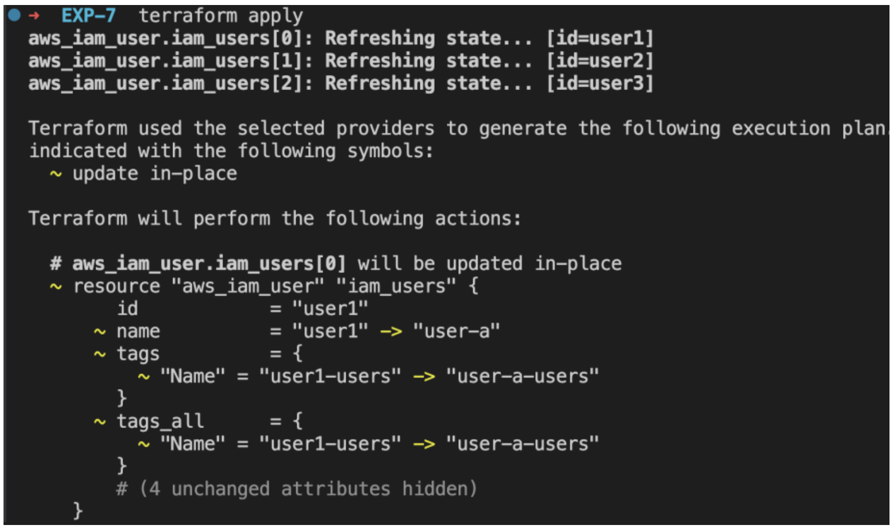
User name	Path	Groups	Last activity	MFA	Password age	Conso
user1	/	0	-	-	-	-
user2	/	0	-	-	-	-
user3	/	0	-	-	-	-

7. **Add or remove IAM Users by changing main.tf file.**

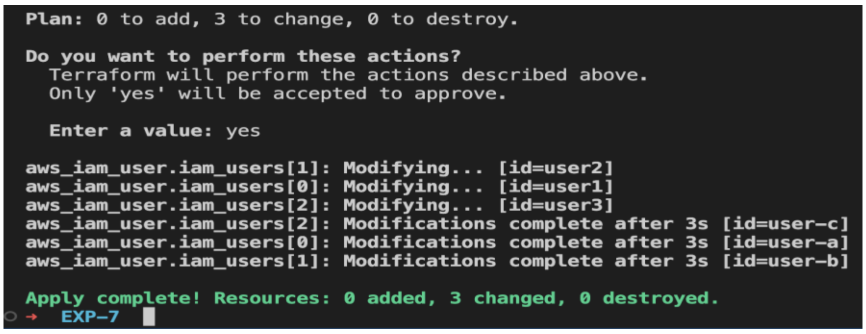


```
EXP-7 > main.tf > variable "iam_users" > [ ] default > 2
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "5.35.0"
    }
  }
}
provider "aws" {
  region     = "ap-south-1"
  access_key = "AKIATG3O3S6EEAAAF6PE"
  secret_key = "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"
}
variable "iam_users" {
  type    = list(string)
  default = ["user-a", "user-b", "user-c"]
}
resource "aws_iam_user" "iam_users" {
  count = length(var.iam_users)
  name  = var.iam_users[count.index]
  tags = {
    Name = "${var.iam_users[count.index]}-users"
  }
}
```

8. **Use terraform apply to apply changes.**

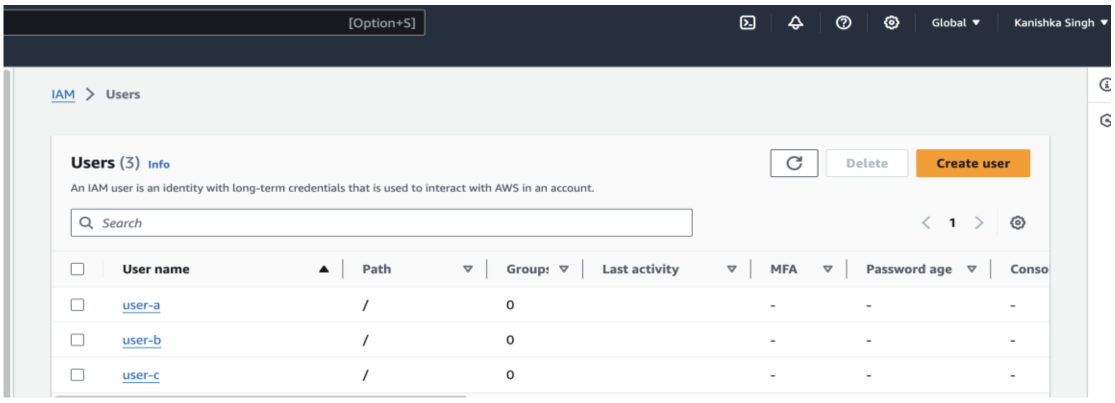


```
➜  EXP-7  terraform apply
aws_iam_user.iam_users[0]: Refreshing state... [id=user1]
aws_iam_user.iam_users[1]: Refreshing state... [id=user2]
aws_iam_user.iam_users[2]: Refreshing state... [id=user3]

Terraform used the selected providers to generate the following execution plan
indicated with the following symbols:
  ~ update in-place

Terraform will perform the following actions:

  # aws_iam_user.iam_users[0] will be updated in-place
  ~ resource "aws_iam_user" "iam_users" {
        id            = "user1"
      ~ name          = "user1" -> "user-a"
      ~ tags          = {
          ~ "Name" = "user1-users" -> "user-a-users"
        }
      ~ tags_all      = {
          ~ "Name" = "user1-users" -> "user-a-users"
        }
        # (4 unchanged attributes hidden)
    }
```



```
Plan: 0 to add, 3 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_iam_user.iam_users[1]: Modifying... [id=user2]
aws_iam_user.iam_users[0]: Modifying... [id=user1]
aws_iam_user.iam_users[2]: Modifying... [id=user3]
aws_iam_user.iam_users[2]: Modifications complete after 3s [id=user-c]
aws_iam_user.iam_users[0]: Modifications complete after 3s [id=user-a]
aws_iam_user.iam_users[1]: Modifications complete after 3s [id=user-b]

Apply complete! Resources: 0 added, 3 changed, 0 destroyed.
➜  EXP-7
```

9. **Verify it using AWS Console.**

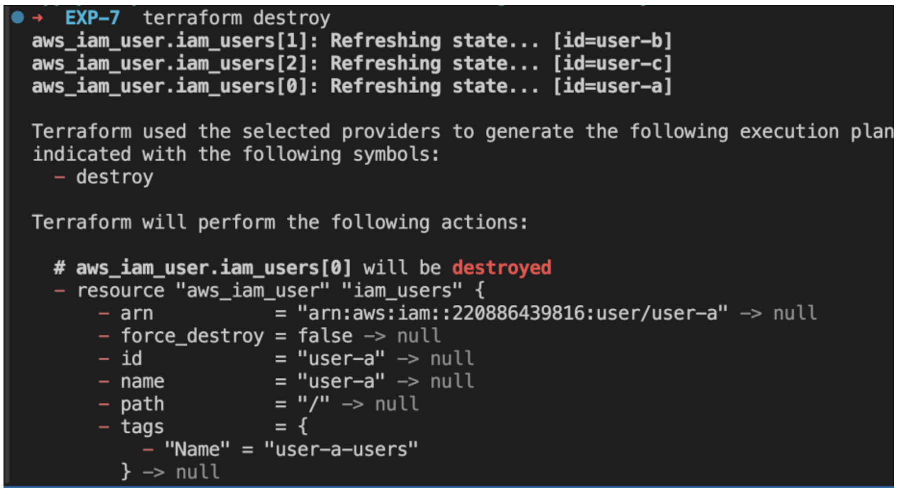


Users (3)

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

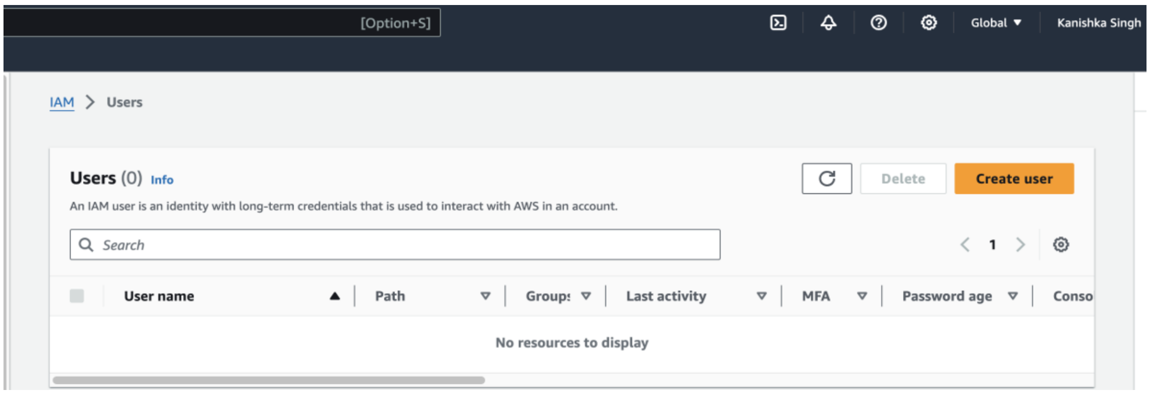
User name	Path	Groups	Last activity	MFA	Password age	Conso
user-a	/	0		-	-	-
user-b	/	0		-	-	-
user-c	/	0		-	-	-

10. **Delete all IAM users using terraform destroy.**



```
● → EXP-7 terraform destroy
aws_iam_user.iam_users[1]: Refreshing state... [id=user-b]
aws_iam_user.iam_users[2]: Refreshing state... [id=user-c]
aws_iam_user.iam_users[0]: Refreshing state... [id=user-a]

Terraform used the selected providers to generate the following execution plan
indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_iam_user.iam_users[0] will be destroyed
  - resource "aws_iam_user" "iam_users" {
      - arn           = "arn:aws:iam::220886439816:user/user-a" -> null
      - force_destroy = false -> null
      - id            = "user-a" -> null
      - name          = "user-a" -> null
      - path          = "/" -> null
      - tags          = {
          - "Name" = "user-a-users"
        } -> null
```

11. **Verify it by AWS Console.**



Users (0)

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

User name	Path	Groups	Last activity	MFA	Password age	Conso

No resources to display